National Technical University of Athens
School of Electrical and Computer Engineering
Forecasting & Strategy Unit

Forecasting Techniques

The R software

Part #2

FSU

Exponential Smoothing in R

```
ses(x, h=10, level=c(80,95),
fan=FALSE,
initial=c("optimal","simple"),
alpha=NULL)
```

```
holt(x, h=10, damped=FALSE,
level=c(80,95), fan=FALSE,
initial=c("optimal", "simple"),
exponential=FALSE, alpha=NULL,
beta=NULL
```

x	a numeric vector or time series
h	Number of periods for forecasting.
damped	If TRUE, use a damped trend.
seasonal	Type of seasonality in hw model. "additive" or "multiplicative"
level	Confidence level for prediction intervals.
fan	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.
initial	Method used for selecting initial state values. If optimal, the initial values are optimized along with the smoothing parameters using ets. If simple, the initial values are set to values obtained using simple calculations on the first few observations. See Hyndman & Athanasopoulos (2012) for details.
exponential	If TRUE, an exponential trend is fitted. Otherwise, the trend is (locally) linear.
alpha	Value of smoothing parameter for the level. If NULL, it will be estimated.
beta	Value of smoothing parameter for the trend. If NULL, it will be estimated.
gamma	Value of smoothing parameter for the seasonal component. If NULL, it will be estimated.
...	Other arguments passed to forecast.ets.

FSU

Exponential Smoothing in R

```
ses(x, h=10, level=c(80,95),
fan=FALSE,
initial=c("optimal","simple"),
alpha=NULL, ...)

holt(x, h=10, damped=FALSE,
level=c(80,95), fan=FALSE,
initial=c("optimal", "simple"),
exponential=FALSE, alpha=NULL,
beta=NULL, ...)
```

An object of class "forecast" is a list containing at least the following elements:

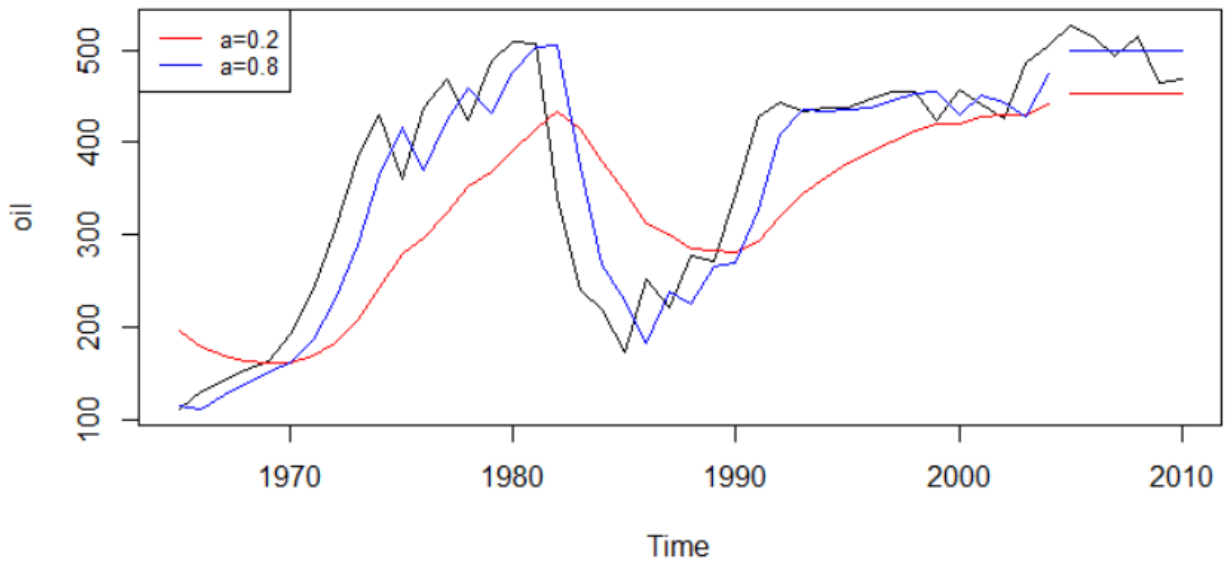
model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

FSU

Exponential Smoothing in R

```
library(forecast)
library(fpp)
oildata_train <- window(oil, start = 1965, end = 2004)
oildata_test <- window(oil, start = 2005, end = 2010)

fit1 <- ses(oildata_train, alpha = 0.2, h=6)
fit2 <- ses(oildata_train, alpha = 0.8, h=6)

plot(oil, type="l")
lines(fit1$fitted, col="red")
lines(fit1$mean, col="red")
lines(fit2$fitted, col="blue")
lines(fit2$mean, col="blue")
legend("topleft", legend=c("a=0.2", "a=0.8"),
       col=c("red", "blue"), lty=1, cex=0.8)
```

Fit and evaluate SES for various smoothing parameters (a)



FSU

Exponential Smoothing in R

```
#Insample accuracy
mse_in1 <- mean((oildata_train - fit1$fitted)^2)
mse_in2 <- mean((oildata_train - fit2$fitted)^2)
#Outsample accuracy
mse_out1 <- mean((oildata_test - fit1$mean)^2)
mse_out2 <- mean((oildata_test - fit2$mean)^2)

c(mse_in1, mse_in2)
c(mse_out1, mse_out2)
```

0.2
0.8
0.2
0.8

```
> c(mse_in1, mse_in2)
[1] 9104.339 3006.520
> c(mse_out1, mse_out2)
[1] 2457.2304  576.7129
```

Although this is not always the case, the model that produced the most accurate forecasts when trained, is also the most accurate model in predicting the future

Fitting the "optimal" SES model

```
#Optimal parameters
model <- ses(oildata_train)
model$model
```

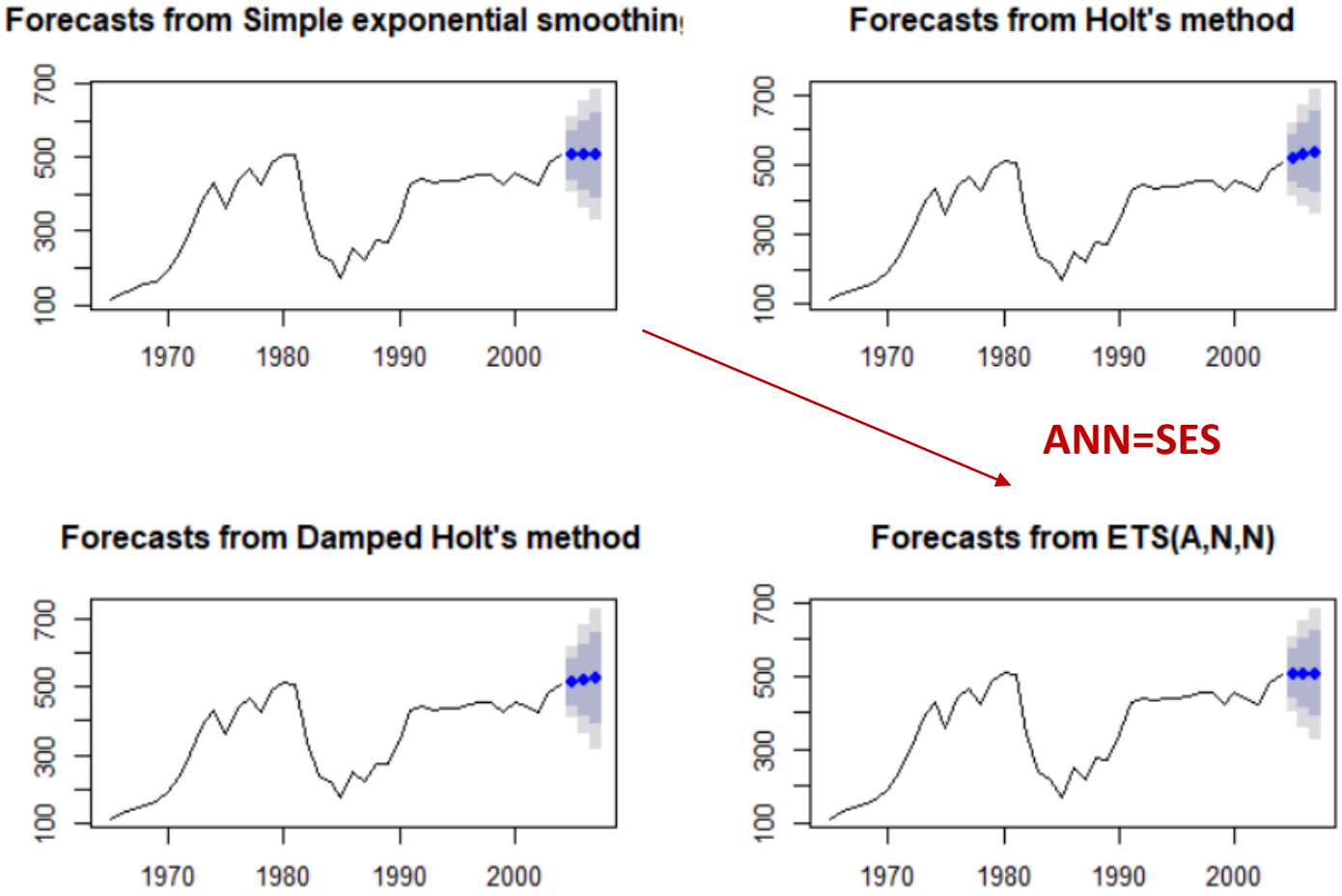
```
Simple exponential smoothing

Call:
 ses(y = oildata_train)

  Smoothing parameters:
    alpha = 0.9999

  Initial states:
    l = 110.8832

  sigma:  52.6202

     AIC     AICc      BIC
468.5515 469.2182 473.6181
```

FSU

Exponential Smoothing in R

```r
#Exponential smoothing models
model1 <- ses(oildata_train, h=3)
model2 <- holt(oildata_train, h=3)
model3 <- holt(oildata_train, damped = TRUE, h=3)
model4 <- forecast(ets(oildata_train), h=3)
par(mfrow=c(2,2))
plot(model1) ; plot(model2)
plot(model3) ; plot(model4)
```

Fit and evaluate different exponential smoothing models



Exponential Smoothing in R

```
#Insample accuracy
mse_in1 <- mean((oildata_train - model1$fitted)^2)
mse_in2 <- mean((oildata_train - model2$fitted)^2)
mse_in3 <- mean((oildata_train - model3$fitted)^2)
mse_in4 <- mean((oildata_train - model4$fitted)^2)
#Outsample accuracy
mse_out1 <- mean((oildata_test - model1$mean)^2)
mse_out2 <- mean((oildata_test - model2$mean)^2)
mse_out3 <- mean((oildata_test - model3$mean)^2)
mse_out4 <- mean((oildata_test - model4$mean)^2)

c(mse_in1, mse_in2, mse_in3, mse_in4)
c(mse_out1, mse_out2, mse_out3, mse_out4)
```

SES Holt Damped ETS

```
> c(mse_in1, mse_in2, mse_in3, mse_in4)
[1] 2630.445 2557.511 2563.338 2630.445
> c(mse_out1, mse_out2, mse_out3, mse_out4)
[1] 212.6920 665.8960 348.9897 212.6920
```

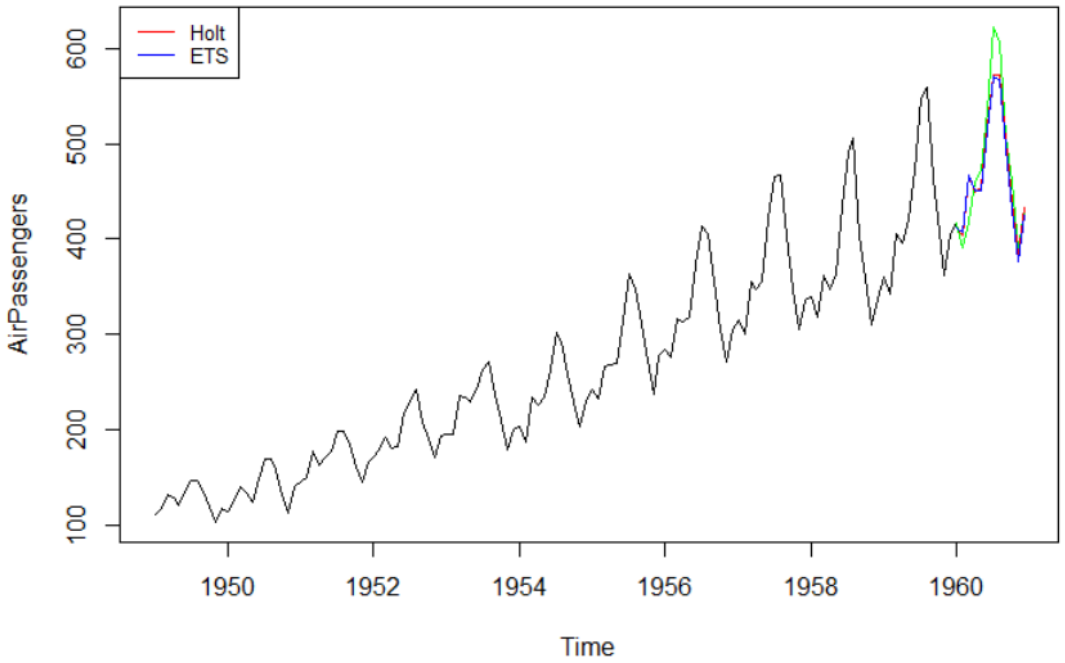
- Holt is considered the most accurate model (based on in-sample data) but SES works much better for the out-of-sample data
- SES is correctly identified by ETS as the most appropriate model for automatic forecasting

FSU

Exponential Smoothing in R

Exponential Smoothing for Seasonal Time Series

```
#Seasonal ts
insample <- window(AirPassengers, start=c(1949,1) , end=c(1959,12))
outsample <- window(AirPassengers, start=c(1960,1) , end=c(1960,12))

SI <- decompose(insample, type="multiplicative")$seasonal
Dy <- insample/SI
frc1 <- holt(Dy, h=12)$mean*as.numeric(tail(SI,12))

frc2 <- forecast(ets(insample), h=12)$mean

plot(AirPassengers)
lines(outsample, col="green")
lines(frc1, col="red")
lines(frc2, col="blue")
legend("topleft", legend=c("Holt", "ETS"),
       col=c("red", "blue"), lty=1, cex=0.8)

mean((outsample - frc1)^2)
mean((outsample - frc2)^2)
```

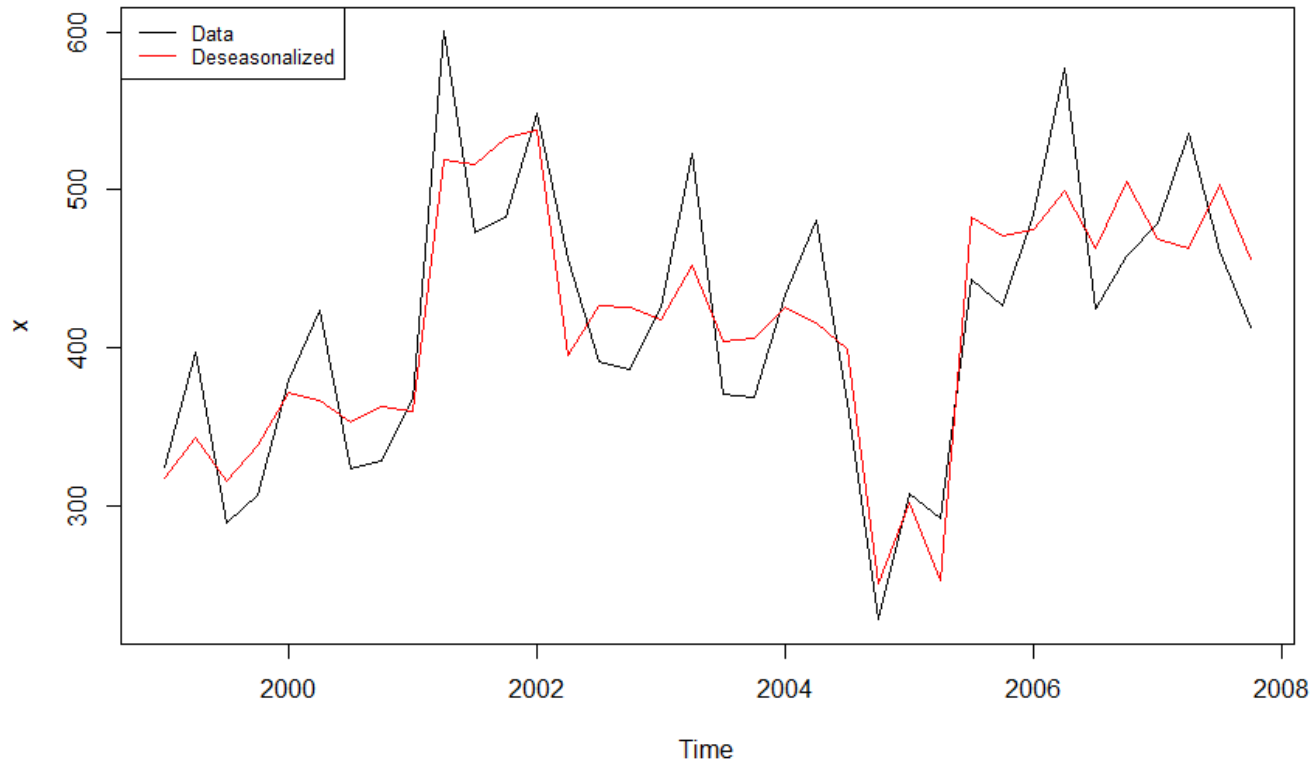



```
> mean((outsample - frc1)^2)
[1] 618.7616
> mean((outsample - frc2)^2)
[1] 750.6526
```

FSU

Special events and actions

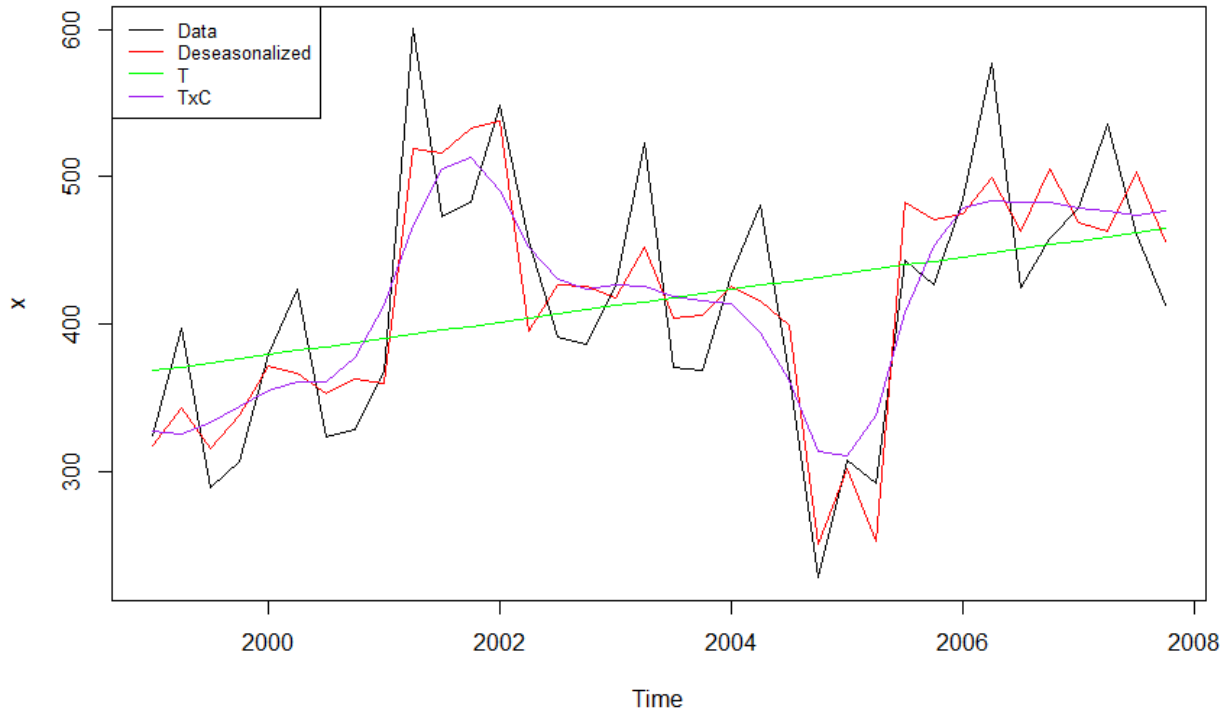
```
library(forecast)

x <- ts(c(324.25,396.82,289.42,307.17,379.36,424.1,323.36,328.91,367.38,
          600.29,473.4,483.13,548.5,456.72,391.18,386.09,426.02,523.13,370.5,
          368.3,433.78,480.79,365.72,227.87,307.47,292.07,443.09,427.04,
          484.69,577.09,424.36,458.45,478.35,535.68,461.52,413.23),
        frequency = 4, start = c(1999,1))

# Estimate KMO4 and LE
KMO_4 <- ma(x,4)
LE <- x/KMO_4

# Estimate SE and deseasonalized series
de <- na.omit(data.frame(rep(c(1:4),9), as.numeric(LE)))
colnames(de) <- c("q","le")
DE <- c()
for (i in 1:4){
  temp <- de[de==i,]$le
  min_Q <- min(temp)
  max_Q <- max(temp)
  DE <- c(DE, mean(temp[(temp>min_Q)&(temp<max_Q)]))
}
DE <- DE/(sum(DE)/4)
DE <- rep(DE,9)
des_x <- x/DE

plot(x)
lines(des_x, col="red", ylab="",xlab="time")
legend("topleft", legend=c("Data", "Deseasonalized"),
       col=c("black", "red"), lty=1, cex=0.8)
```



FSU

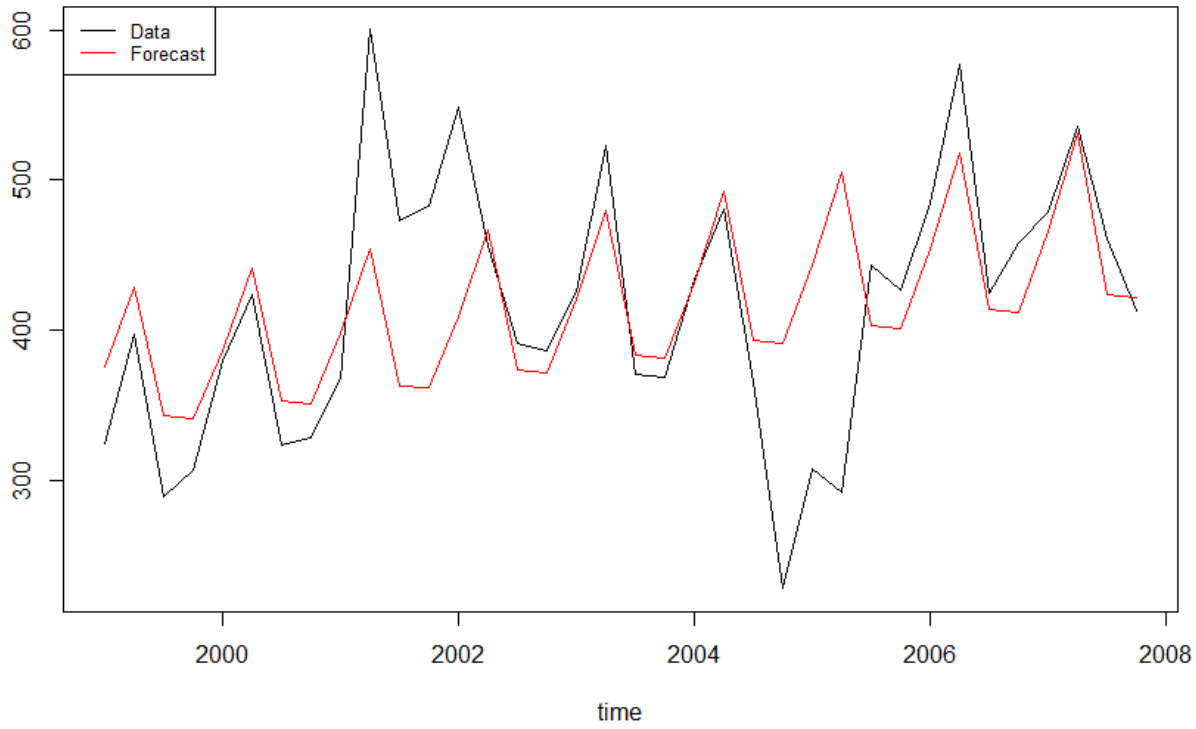
Special events and actions

```
# Estimate KMO3
KMO_3 <- ma(des_x,3)
KKMO_3 <- ma(KMO_3,3)
KKMO_3[2] <- KMO_3[2] #Fill missing values
KKMO_3[length(KKMO_3)-1] <- KMO_3[length(KKMO_3)-1]
KKMO_3[1] <- (des_x[1]+des_x[2]+KMO_3[2]-KMO_3[3])/2
KKMO_3[length(KKMO_3)] <- (des_x[length(KKMO_3)]+des_x[length(KKMO_3)-1]+
                                    KMO_3[length(KKMO_3)-1]-KMO_3[length(KKMO_3)-2])/2

# Estimate trend
xx <- c(1:length(x))
yy <- KKMO_3
lrl <- lm(yy~xx)
summary(lrl)
Trend <- ts(as.numeric(predict(lrl)), frequency = 4, start = c(1999,1))

plot(x)
lines(des_x, col="red", ylab="",xlab="time")
lines(Trend, col="green", ylab="",xlab="time")
lines(KKMO_3, col="purple", ylab="",xlab="time")
legend("topleft", legend=c("Data", "Deseasonalized", "T", "TxC"),
      col=c("black", "red", "green", "purple"), lty=1, cex=0.8)
```



Special events and actions

```
# Method 2 - Forecasts based on decomposition
FM <- Trend*DE
plot(x, ylab="",xlab="time")
lines(FM, col="red")
legend("topleft", legend=c("Data", "Forecast"),
       col=c("black", "red"), lty=1, cex=0.8)
stD <- sd(FM)
Dmean <- mean(des_x)

# Method 1
ratio_m1_1 <- des_x/KKMO_3
ratio_m1_2 <- des_x/FM

# Method 3 - KMO5 and KMO7
KMO5 <- ma(des_x,5)
KMO7 <- ma(des_x,7)
ratio_m3 <- KMO7/KMO5

# Method 4
ratio_m4 <- des_x/KMO_4
```



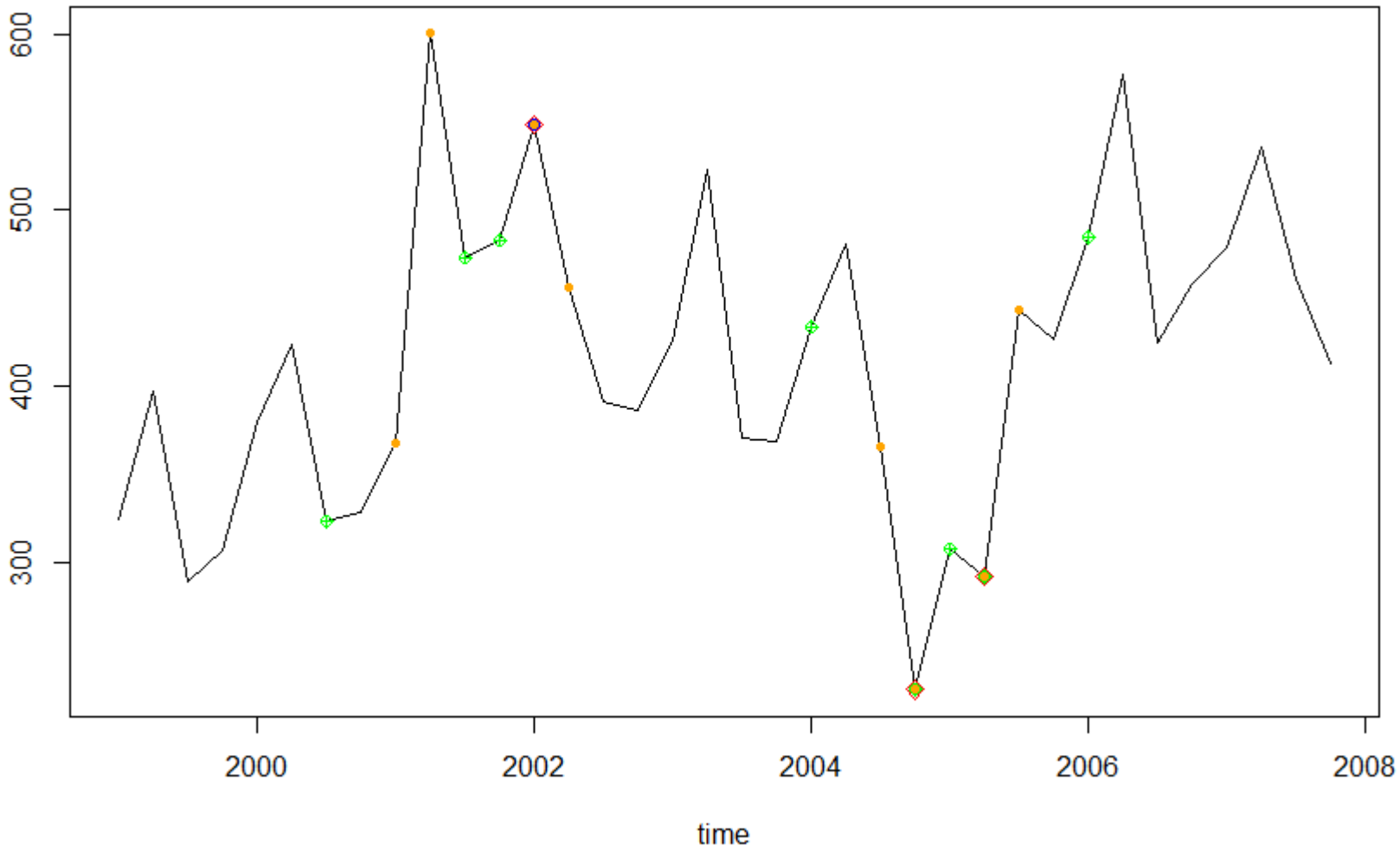
Special events and actions

```
# Apply method 1
criteria_1 <- data.frame(ratio_m1_1, ratio_m1_2)
ta = 2 ; tb = 5
criteria_1$is_r1 = criteria_1$is_r2 <- 0
criteria_1[criteria_1$ratio_m1_1 >= 1.1-ta/100,]$is_r1 <- 1
criteria_1[criteria_1$ratio_m1_1 <= 0.9+ta/100,]$is_r1 <- 1
criteria_1[criteria_1$ratio_m1_2 >= 1.25-tb/100,]$is_r2 <- 1
criteria_1[criteria_1$ratio_m1_2 <= 0.75+tb/100,]$is_r2 <- 1
criteria_1$is <- criteria_1$is_r1 * criteria_1$is_r2
criteria_1$data <- x
criteria_1[criteria_1$is==0,]$data <- NA
plot(x, ylab="",xlab="time")
points(ts(criteria_1$data, frequency = 4, start = c(1999,1)), col="red", pch = 5)

# Apply method 2
t = 0.6
upper <- Dmean + (3-t)*stD
lower <- Dmean - (3-t)*stD
criteria_2 <- data.frame(des_x, x)
criteria_2$is_r <- 0
criteria_2[(criteria_2$des_x >= upper)|(criteria_2$des_x <= lower),]$is_r <- 1
criteria_2[criteria_2$is_r==0,]$x <- NA
points(ts(criteria_2$x, frequency = 4, start = c(1999,1)), col="blue", pch = 19)
```

```
# Apply method 3
t = 1
criteria_3 <- data.frame(des_x, x, ratio_m3)
criteria_3$is_r <- 0
criteria_3[is.na(criteria_3$ratio_m3),]$ratio_m3 <- 1
criteria_3[(criteria_3$ratio_m3 >= (1.05-t/100))|(criteria_3$ratio_m3 <=
(0.95+t/100)),]$is_r <- 1
criteria_3[criteria_3$is_r==0,]$x <- NA
points(ts(criteria_3$x, frequency = 4, start = c(1999,1)), col="green", pch = 10)

# Apply method 4
t = 2
criteria_4 <- data.frame(des_x, x, ratio_m4)
criteria_4$is_r <- 0
criteria_4[is.na(criteria_4$ratio_m4),]$ratio_m4 <- 1
criteria_4[(criteria_4$ratio_m4 >= (1.1-t/100))|(criteria_4$ratio_m4 <=
(0.9+t/100)),]$is_r <- 1
criteria_4[criteria_4$is_r==0,]$x <- NA
points(ts(criteria_4$x, frequency = 4, start = c(1999,1)), col="orange", pch = 20)
```

FSU

Special events and actions



Special events and actions

```
#Combine results
criteria <- data.frame(criteria_1$is, criteria_2$is_r, criteria_3$is_r, criteria_1$is)
colnames(criteria) <- c("m1", "m2", "m3", "m4")
criteria$SE <- criteria$m1+criteria$m2+criteria$m3+criteria$m4
criteria$data <- x
criteria$data_adj <- des_x
criteria[criteria$SE<2,]$data <- NA
plot(x, ylab="",xlab="time")
points(ts(criteria$data, frequency = 4, start = c(1999,1)), col="red", pch = 5)

for (i in 2:nrow(criteria)){
  if (is.na(criteria$data[i])==F){
    criteria$data_adj[i] <- (criteria$data_adj[i-1] + criteria$data_adj[i+1])/2
  }
}
Adjusted <- ts(criteria$data_adj*DE, frequency = 4, start = c(1999,1))
Impact <- (des_x-criteria$data_adj)*100/criteria$data_adj
Impact[Impact!=0]

plot(x, ylab="",xlab="time")
lines(Adjusted, col="red")
legend("topleft", legend=c("Data", "Adjusted"),
       col=c("black", "red"), lty=1, cex=0.8)
```